Search Bar

The <aava-search-bar> component provides a comprehensive search input solution with integrated search and send icons. Built on top of the textbox component, it offers consistent styling, multiple size variants, and flexible configuration options for various search scenarios.

How to use

Basic Usage

Simple search bar implementation with default settings and integrated search functionality.

```
import { AavaSearchBarComponent } from "@aava/play-core";
```

Basic Features

- Integrated Icons : Built-in search, send and close icons for intuitive user experience
- Textbox Foundation : Inherits all textbox functionality and styling
- Search Events : Emits search events when user click the send icon
- Responsive Design : Adapts to different screen sizes and container widths
- Accessibility : Full keyboard navigation and screen reader support

Variants

Simple search bar implementation with default settings and integrated search functionality.

```
&lt;aava-search-bar
  placeholder="Enter your search term..."
  [closeButton]="true"
  (searchClick)="onSearchClick($event)"
  (searchChange)="onSearchChange($event)"
  (onClose)="onClose($event)"
&gt;&lt;/aava-search-bar&gt;
```

Available Variants

- closeButtonVisibility : Controls the visibility behavior of the close (clear) button. This input determines when the close button should be displayed based on the textbox content or user preference.
- sendButton : Controls the visibility of the send button.

- searchIconColor : Controls the search icon color
- sendIconColor : Controls the send icon color

Size

Four size variants to accommodate different interface densities and design requirements.

```
onSearchClick(searchTerm: string) {
  console.log('Search clicked with term:', searchTerm);
}
onSearchChange(event: Event) {
  console.log('Textbox changed:', this.currentInput);
}
onClose(searchTerm: string) {
  console.log('Clear search:', searchTerm);
}
```

Available Sizes

- xs (Extra Small) - Compact size for minimal interfaces (16px icon)
- md (Medium) - Standard size for most search scenarios (20px icon, default)
- lg (Large) - Prominent size for important search interfaces (24px icon)
- xl (Extra Large) - Very prominent size for high-visibility search (24px icon)

Size Features

- Proportional Scaling : Icon sizes scale appropriately with component size
- Consistent Spacing : Maintains proper proportions across all size variants
- Touch Targets : All sizes meet minimum touch target requirements
- Visual Hierarchy : Larger sizes provide better emphasis for primary search areas

States

Different interaction states for various user scenarios and accessibility requirements.

```
&lt;aava-search-bar closeButtonVisibility="always"&gt;&lt;/aava-search-bar&gt;
&lt;aava-search-bar closeButtonVisibility="hidden"&gt;&lt;/aava-search-bar&gt;
&lt;aava-search-bar
  closeButtonVisibility="always"
  [sendButton]="false"
&gt;&lt;/aava-search-bar&gt;
&lt;aava-search-bar [sendButton]="false"&gt;&lt;/aava-search-bar&gt;

&lt;aava-search-bar
  placeholder="Primary"
  searchIconColor="#3b82f6"
  sendIconColor="#3b82f6"
&gt;&lt;/aava-search-bar&gt;

&lt;aava-search-bar
  placeholder="Success"
  searchIconColor="#10b981"
  sendIconColor="#10b981"
&gt;&lt;/aava-search-bar&gt;
&lt;aava-search-bar
  placeholder="Warning"
  searchIconColor="#f59e0b"
  sendIconColor="#f59e0b"
&gt;&lt;/aava-search-bar&gt;
&lt;aava-search-bar
  placeholder="Danger"
  searchIconColor="#ef4444"
  sendIconColor="#ef4444"
&gt;&lt;/aava-search-bar&gt;
```

State Features

- Default State : Normal interaction with full functionality
- Disabled State : Non-interactive state with visual feedback
- Focus State : Clear focus indicators for keyboard navigation
- Hover State : Subtle hover effects for interactive elements

API Reference

Inputs

Property	Type	Default	Description
label	string?	''	Label text for the search input
disabled	boolean	false	Whether the search bar is disabled

Property	Type	Default	Description
variant	string?	''	Visual variant of the search bar
size	'xs' | 'md' | 'lg' | 'xl'	'md'	Size variant of the search bar
serachIconColor	string	'#000000'	Color for the search icon
sendIconColor	string	'#000000'	Color for the send icon
sendButton	boolean	true	Whether the send button is shown
closeButtonVisibility	'auto' | 'always' | 'hidden'	hidden	Close button visibility

Outputs

Event	Type	Description
searchClick	EventEmitter<string>	Emitted when user clicks the send icon
searchChange	EventEmitter<string>	Emitted when the search input value changes
onClose	EventEmitter<string>	Emitted when the close button is clicked

Best Practices

Design Guidelines

- Clear Visual Hierarchy : Use appropriate sizes for primary vs. secondary search areas
- Consistent Iconography : Maintain consistent icon colors across your application
- Accessible Contrast : Ensure sufficient contrast between icons and backgrounds
- Responsive Sizing : Choose sizes that work well on all target devices
- Brand Consistency : Use icon colors that align with your brand guidelines

Accessibility

- Keyboard Navigation : Ensure full keyboard accessibility for all interactions
- Screen Reader Support : Provide clear labels and descriptions for search functionality
- Focus Management : Maintain clear focus indicators for all interactive elements
- ARIA Labels : Use appropriate ARIA attributes for search context
- Color Independence : Don't rely solely on color to convey information

Performance

- Event Handling : Debounce search input events for better performance
- Icon Optimization : Use optimized SVG icons for consistent rendering
- Change Detection : Leverage OnPush strategy for optimal performance
- Memory Management : Clean up event listeners and subscriptions properly
- Bundle Size : Import only needed components to minimize bundle size

User Experience

- Clear Purpose : Make it obvious that this is a search input
- Visual Feedback : Provide immediate feedback for user interactions
- Search Suggestions : Consider adding autocomplete or search suggestions
- Loading States : Show loading indicators during search operations
- Error Handling : Provide clear error messages for failed searches

Implementation Considerations

- State Management : Properly manage search state in your application
- Search Logic : Implement efficient search algorithms and data structures
- API Integration : Handle search API calls with proper error handling
- Caching : Cache search results for better performance
- Analytics : Track search usage for user behavior insights

Search Functionality

- Real-time Search : Consider implementing search-as-you-type functionality
- Search History : Maintain search history for user convenience
- Search Filters : Add filtering options for refined search results
- Search Results : Design clear and organized search result displays
- Search Analytics : Monitor search patterns and popular queries

Technical Notes

Component Architecture

The search bar component extends the textbox component:

- Inheritance : Extends AvaTextboxComponent for consistent behavior
- Composition : Uses IconComponent for search and send icons
- Change Detection : Implements OnPush strategy for optimal performance
- Styling : Inherits textbox styling with search-specific customizations

Icon System

The component implements a flexible icon system:

- Dynamic Sizing : Icon sizes are computed based on component size
- Color Management : Handles disabled states with CSS variable fallbacks
- Slot System : Uses Angular slot system for icon positioning
- Interactive Elements : Send icon is clickable with proper cursor styling

Event Handling

The component manages multiple event types:

- Input Changes : Tracks search value changes in real-time
- Search Triggers : Emits search events when send icon is clicked
- Textbox Events : Inherits all textbox event handling
- Icon Interactions : Handles icon-specific interactions and styling

Size Mapping

The component maps size variants to icon dimensions:

- XS : 16px icons for compact interfaces
- MD : 20px icons for standard usage (default)
- LG : 24px icons for prominent interfaces
- XL : 24px icons for high-visibility areas

CSS Integration

The component integrates with the design system:

- CSS Variables : Uses semantic CSS variables for theming
- Responsive Design : Adapts to different screen sizes
- State Management : Handles various component states
- Accessibility : Maintains proper contrast and focus indicators